

Forum:[Hack Forums](#)

Thread: [Bypass Windows Defender with Hellsgate: Introducing HellshadeGate](#)

kitsuneshade - August 14, 2025, 3:26 am

Bypass Windows Defender with Hellsgate: Introducing HellshadeGate

Overview

HellshadeGate is a basic, modular Rust loader designed to demonstrate direct syscall techniques for evading Windows Defender and other user-mode security solutions. By leveraging the Hellsgate approach, HellshadeGate dynamically resolves syscall numbers at runtime and executes shellcode using direct syscalls, bypassing common hooks and detection mechanisms.

GitHub: [hXXps://github.com/kitsuneshade/HellshadeGate](#)

How Does It Work?

Direct Syscalls:

HellshadeGate uses a custom x64 assembly stub to invoke Windows syscalls directly, avoiding API hooks set by EDRs and Defender.

Dynamic Syscall Resolution:

The loader parses the export table of `ntdll.dll` at runtime, extracting syscall numbers and building a syscall table for critical functions like `NtAllocateVirtualMemory`, `NtWriteVirtualMemory`, `NtProtectVirtualMemory`, and `NtCreateThreadEx`.

Shellcode Execution:

Reads base64-encoded shellcode, allocates RWX memory, writes the shellcode, changes protection, and executes it in a new thread—all via direct syscalls.

Why Is This Effective?

Bypasses User-Mode Hooks:

Windows Defender and most EDRs hook user-mode APIs (like `VirtualAlloc`, `CreateThread`). HellshadeGate calls syscalls directly, skipping these hooks.

No Hardcoded Syscall Numbers:

Syscall numbers are resolved dynamically, making static analysis and signature-based detection harder.

Minimal Footprint:

The loader is simple and modular, making it easy to audit, extend, or integrate into other projects.

How to Improve HellshadeGate

HellshadeGate is a basic loader, but it can be extended in many ways:

Add More Syscalls:

Expand the syscall table to support process injection, remote thread creation, or advanced memory operations.

Obfuscate Shellcode Loading:

Implement encryption or custom encoding for shellcode, decrypting it at runtime.

Integrate Network Capabilities:

Download shellcode from a remote server or C2 channel.

Stealth Enhancements:

Use techniques like unhooking `ntdll.dll`, process hollowing, or indirect syscalls for even greater evasion.

Multi-Architecture Support:

Add support for x86 or ARM architectures.

Example Usage

Generate your shellcode (e.g., with msfvenom), encode it in base64, and save as shellcode.txt.
Build the project:

```
cargo build --release
```

Run the loader and observe the debug output.

Disclaimer

This project is for educational and research purposes only. Use responsibly and only in environments where you have explicit permission.

Bypass Windows Defender with Hellsgate—try HellshadeGate and start exploring advanced evasion techniques!

DramaKing - August 14, 2025, 4:49 am

No way direct syscalls would avoid detection, and everyone knows that base64 encoding is ineffective.

You're just getting skids' hopes up.

kitsuneshade - August 15, 2025, 1:47 pm

(Aug 14, 2025 04:49 AM)DramaKing Wrote: [\[CITE\]](#) No way direct syscalls would avoid detection, and everyone knows that base64 encoding is ineffective.

You're just getting skids' hopes up.

Hi! Just to clarify: a loader by itself doesn't automatically perform a bypass of security solutions like Windows Defender. The loader is a tool that prepares the environment and executes your payload in a stealthy way, but the actual bypass depends on the techniques you implement (such as direct syscalls, AMSI patching, or fragmenting the shellcode).

In my previous post, I shared a basic resource and initial code that can help you start building a loader capable of bypassing AV/EDR. You'll need to integrate specific evasion techniques (like Hellsgate, Dlnvoke, or syscall stomping) to achieve a real bypass.

??????????? - August 15, 2025, 2:02 pm

(Aug 14, 2025 03:26 AM)kitsuneshade Wrote: [\[CITE\]](#) Bypass Windows Defender with Hellsgate: Introducing HellshadeGate

Overview

HellshadeGate is a basic, modular Rust loader designed to demonstrate direct syscall techniques for evading Windows Defender and other user-mode security solutions. By leveraging the Hellsgate approach, HellshadeGate dynamically resolves syscall numbers at runtime and executes shellcode using direct syscalls, bypassing common hooks and detection mechanisms.

GitHub: [hXXps: //github . com / kitsuneshade / HellshadeGate](https://github.com/kitsuneshade/HellshadeGate)

How Does It Work?

Direct Syscalls:

HellshadeGate uses a custom x64 assembly stub to invoke Windows syscalls directly, avoiding API hooks set by EDRs and Defender.

Dynamic Syscall Resolution:

The loader parses the export table of ntdll.dll at runtime, extracting syscall numbers and building a syscall table for critical functions like NtAllocateVirtualMemory, NtWriteVirtualMemory, NtProtectVirtualMemory, and NtCreateThreadEx.

Shellcode Execution:

Reads base64-encoded shellcode, allocates RWX memory, writes the shellcode, changes

protection, and executes it in a new thread—all via direct syscalls.

Why Is This Effective?

Bypasses User-Mode Hooks:

Windows Defender and most EDRs hook user-mode APIs (like VirtualAlloc, CreateThread).

HellshadeGate calls syscalls directly, skipping these hooks.

No Hardcoded Syscall Numbers:

Syscall numbers are resolved dynamically, making static analysis and signature-based detection harder.

Minimal Footprint:

The loader is simple and modular, making it easy to audit, extend, or integrate into other projects.

How to Improve HellshadeGate

HellshadeGate is a basic loader, but it can be extended in many ways:

Add More Syscalls:

Expand the syscall table to support process injection, remote thread creation, or advanced memory operations.

Obfuscate Shellcode Loading:

Implement encryption or custom encoding for shellcode, decrypting it at runtime.

Integrate Network Capabilities:

Download shellcode from a remote server or C2 channel.

Stealth Enhancements:

Use techniques like unhooking ntdll.dll, process hollowing, or indirect syscalls for even greater evasion.

Multi-Architecture Support:

Add support for x86 or ARM architectures.

Example Usage

Generate your shellcode (e.g., with msfvenom), encode it in base64, and save as shellcode.txt.

Build the project:

```
cargo build --release
```

Run the loader and observe the debug output.

Disclaimer

This project is for educational and research purposes only. Use responsibly and only in environments where you have explicit permission.

Bypass Windows Defender with Hellsgate—try HellshadeGate and start exploring advanced evasion techniques!

HellshadeGate might dodge old-school checks, but modern Windows Defender watches much deeper and spots the same shady patterns anyway. Even if it sneaks past the first layer, the newer defenses will likely catch it fast.

kitsuneshade - August 15, 2025, 5:14 pm

(Aug 15, 2025 02:02 PM)??????? Wrote: [\[CITE\]](#) HellshadeGate might dodge old-school checks, but modern Windows Defender watches much deeper and spots the same shady patterns anyway. Even if it sneaks past the first layer, the newer defenses will likely catch it fast.

HellshadeGate provides a solid foundation with direct syscalls (Hell's Gate) and dynamic obfuscation, but modern EDRs (Defender ATP, CrowdStrike) require deeper evasion.

Key Upgrades for EDR Evasion:

Syscall Stack Spoofing – Use ROP chains + SysWhispers3 to break stack trace analysis.

Early Bird APC + Thread Hijacking – Target low-telemetry processes (e.g., werfault.exe) to avoid behavioral detection.

Hardware Breakpoints + VEH Execution – Run shellcode via Vectored Exception Handling (VEH) to bypass API hooks.

Kernel Callback Wiping – Disable PspCreateThreadNotifyRoutine (via driver exploit or kernel R/W) to hide injected threads.

Dynamic Syscall ID Obfuscation – Modify SSNs at runtime using SSN sorting to evade static syscall detection.

Integration into HellshadeGate:

Replace Nt*/Zw* calls with runtime-generated syscalls (SSN brute-forcing or hash-based lookup).

Implement memfd-backed payloads (anonymous memory mapping) to avoid fileless detection.

Patch ETW in userland (hook EtwEventWrite or disable via NtTraceEvent syscall).

Sleep Masking + Timer Queue Evasion – Use WaitForSingleObjectEx with obfuscated delays to bypass sandboxed execution checks.

Next-Level Stealth:

Indirect Syscalls via Return Address Spoofing (mimic legit call stacks).

Kernel-Mode Unhooking (if driver access is possible) to disable EDR's kernel callbacks.

By integrating these techniques, HellshadeGate can bypass memory scans, behavioral analysis, and kernel telemetry while maintaining operational stealth.

???????????? - August 16, 2025, 8:45 am

(Aug 15, 2025 05:14 PM)kitsuneshade Wrote: [CITE](#) HellshadeGate provides a solid foundation with direct syscalls (Hell's Gate) and dynamic obfuscation, but modern EDRs (Defender ATP, CrowdStrike) require deeper evasion.

Key Upgrades for EDR Evasion:

Syscall Stack Spoofing – Use ROP chains + SysWhispers3 to break stack trace analysis.

Early Bird APC + Thread Hijacking – Target low-telemetry processes (e.g., werfault.exe) to avoid behavioral detection.

Hardware Breakpoints + VEH Execution – Run shellcode via Vectored Exception Handling (VEH) to bypass API hooks.

Kernel Callback Wiping – Disable PspCreateThreadNotifyRoutine (via driver exploit or kernel R/W) to hide injected threads.

Dynamic Syscall ID Obfuscation – Modify SSNs at runtime using SSN sorting to evade static syscall detection.

Integration into HellshadeGate:

Replace Nt*/Zw* calls with runtime-generated syscalls (SSN brute-forcing or hash-based lookup).

Implement memfd-backed payloads (anonymous memory mapping) to avoid fileless detection.

Patch ETW in userland (hook EtwEventWrite or disable via NtTraceEvent syscall).

Sleep Masking + Timer Queue Evasion – Use WaitForSingleObjectEx with obfuscated delays to bypass sandboxed execution checks.

Next-Level Stealth:

Indirect Syscalls via Return Address Spoofing (mimic legit call stacks).

Kernel-Mode Unhooking (if driver access is possible) to disable EDR's kernel callbacks.

By integrating these techniques, HellshadeGate can bypass memory scans, behavioral analysis, and kernel telemetry while maintaining operational stealth.

If you are so confident can you do a POC?

kitsuneshade - August 17, 2025, 12:37 am

(Aug 16, 2025 08:45 AM)??????? Wrote: [\[CITE\]](#) If you are so confident can you do a POC?
hxxps:// streamable.com/twd7zv?src=player-page-share

???????????? - August 17, 2025, 5:57 am

(Aug 17, 2025 12:37 AM)kitsuneshade Wrote: [\[CITE\]](#) hxxps://
streamable.com/twd7zv?src=player-page-share
I will check once I am on pc and update you here.

hanes209 - September 18, 2025, 2:23 am

(Aug 17, 2025 12:37 AM)kitsuneshade Wrote: [\[CITE\]](#) hxxps://
streamable.com/twd7zv?src=player-page-share
Video not available anymore